***DRAFT***

# Project Plan on Object Storage Devices

Michael Ernst
*ernst@fnal.gov*

# Table of Contents

# Executive Summary

This project plan represents the first phase of a longer term, three-year project proposal focused on Object Storage Devices (OSD). The xx-month goal of this research project is to develop a demonstration OSD-based storage system that can meet the storage requirements of certain FNAL application areas, specifically those for the Compact Muon Solenoid (CMS) experiment. This storage system will be based on the Panasas Active Scale File System technology (more). The results of this research will provide a basis for assessing the near term and long term capabilities of OSD technologies, and perhaps steer this emerging technology in a way that would benefit future HEP data systems.

This proposal describes the project plan necessary to get to a demonstrable system in 12 months. The components of this plan include:

- Develop an understanding of the CMS storage and define data requirements for a demonstration system
- Demonstration System Design and Construction – Design and build a demonstration system based on the requirements
- Demonstration System Characterization – Run the demonstration system in a small-scale production mode, investigating its features in terms of performance and functionality
- Capabilities Comparison – Compare the strengths and weaknesses of current block-based and NAS-based storage systems to those of a mature OSD system.
- Improvement Analysis – Analyze constraints of current systems that OSD can potentially relax by using a prototype system to be constructed at FNAL for this purpose. Summarize potential improvements of mature OSD systems, but unavailable for study in the prototype.
- Production Scale Analysis – Write a report on what it would take to scale the demonstration system up to a full-scale production system.

This project then leads into the next phase of the overall goal of integrating such a system into the FNAL hardware and software infrastructures and full-scale deployment. However, integration and full-scale deployment is beyond the scope of this specific project.

# Problem Description

FNAL is dealing with a tremendous growth in data storage and retrieval in support of numerous HEP experiments. One such experiment is the Compact Muon Solenoid (CMS) experiment designed to explore the full range of physics at the high-energy frontier up to TeV mass/energy scales made available for the first time at CERN's Large Hadron Collider (LHC). The CMS design stresses precision measurements of electrons, photons and muons, in combination with jets and missing energy.

The high energy and luminosity of the LHC, which are key to its potential for new physics discoveries, lead to technical challenges in computing, data storage and access. The scale, complexity and worldwide geographical spread of the CMS computing and data analysis problems are unprecedented in scientific research. CMS foresees a recorded raw data rate of 1 PB/year (or 100 MB/s during running) at the start of LHC operation. This is the rate of data collected and stored for further analysis, after online filtering by a factor of several hundred thousand, and online processing and data compaction. As the program progresses, it is expected that the combined raw and processed data of the experiment will reach 3 PB by 2007 and 30 PB by roughly 2010 growing at a rate of ~10 PB/year. The anticipated capacity on random access storage devices (e.g. disk) will be as high as 1 PB at the start of the program, depending on the progress of the rapidly evolving technology even higher.

**Physics analysis**

CMS has a very high event rate of 40,000,000 events per second because, like in many HEP experiments, the most important phenomena that the physicists want to observe will occur with only very low probability in a single event. Of the total amount of events per second only some 100 are selected for storage and later analysis. The raw detector data of each selected event is stored as a set of data products, which we consider as a piece of self-contained data, also called 'objects'.

Data analysis on selected data sets is a highly concurrent effort that is undertaken by hundreds of physicists around the globe. Given the vast amount of data to be analyzed and the size of the collaboration (~2000 physicists) the analysis effort will be carried out by utilizing a multi tier hierarchy of distributed, grid enabled computing facilities following the idea of making resources available to members of a particular community regardless where they are installed.

While the raw data will likely stay on a tertiary storage system (robotic tape store or a more advanced inexpensive storage technology) at CERN collections of reconstructed data products will be widely replicated.

While most of the before mentioned data reduction takes place in the online systems the rest is done in an interactive way using the CMS offline computing facilities. Physics analysis on the offline system is an iterative, collaborative process, in which subsequent versions of event feature extraction algorithms and event selection functions are refined until their effects are well understood.

**Data handling problems**
Some of the main points with respect to physics data analysis problems include:

- Data volumes are huge, but the size of an individual data product or object is relatively small (1KB-1MB).
- Workloads are dominated by reading.
- Because of the large data reduction factor in several important types of CMS physics analysis, the input datasets in such analysis efforts will represent very sparse subsets of the set of events over a period of detector running. The sparseness increases as the analysis effort progresses. This has an important impact on data handling in a sense that a traditional system using a fixed partitioning of data product values (physical representation of the value of a data product) over large files, and then stages all data needed by a job by staging all the large files containing that data, will be too inefficient. To achieve the desired efficiency for the CMS workload, it is necessary to extract a sparse subset of the data product values from large datasets, which typically requires a copy operation today. Much more efficient, however, would be direct access to the required objects avoiding the need to walk through massive amounts of irrelevant data.
- In contrast to our applications manipulating data sets with KB to MB size objects, mass storage devices (disk and tape) work most efficiently for large files in the MB to GB range. Reflecting these device characteristics, today's mass storage systems typically have a file level granularity. If there were systems offering object granularity, the application programmer could just store and retrieve objects rather than files. This could be built on top of traditional systems consisting of file systems and block level storage devices, there are however severe implications in terms of some orders of magnitude larger size and complexity of indexing and scheduling tasks associated with managing objects rather than files. In addition a dangerous effect arises from data fragmentation on the storage devices.

**Storage and handling of data product values**
Data product values are always stored in files. Usually there are many related data product values in a single file. Reading and writing data product values to and from files is handled by an object persistency layer inside the CMS executables.

Today, file access by the CMS object persistency layer is always performed on physical files via the UNIX filesystem interface, using regular POSIX I/O calls. These calls could be done directly from the executable containing the CMS physics algorithm on a filesystem mounted on the machine the executable runs on. Note that the mounted filesystem is not necessarily a local disk connected to that machine, it might need to be a much larger filesystem local to the CMS computing facility but not local to the machine. The required service is usually provided by a distributed file system running on a huge amount of network-attached components (e.g. file servers, disks etc). In particular, there are two widespread concepts around today, Storage Area Networks (SAN) and Network Attached Storage (NAS). Implementations of both concepts have inherent limitations in

terms of cross platform interoperability, performance, extendibility over the WAN, security etc. SANs in particular allow high performance, low latency discrete block-level access to storage but there is clearly a lack of solid global file system implementations allowing data sharing among heterogeneous machines, leaving users to deal with a low abstraction level (blocks). SANs are implemented on top of Fibre Channel today, a complex and rather expensive technology. On the contrary NAS allows applications to operate on the file level that, however, comes with a significant performance penalty due to queuing delays on the file server, TCP/IP networking overhead and lack of aggregation mechanisms (e.g. RAID). Current practice at FNAL favors NAS.

Whether we choose to accept the penalties inherent to SANs or NAS one major drawback, common to both solutions, is the fact that a file system is required, an associated abstraction level which doesn't match the characteristics of physics data objects. Another drawback of traditional file systems is that the machine hosting it wants to manage the storage resource (e.g. disk) on a microscopic level (e.g. block) without knowing about the specific characteristics of that particular device. This is not only extremely burdening for the host, there are also severe implications in terms of performance and capacity degradation. An ideal alternative would be to leave this task to the device itself. Talking about efficiency we could even go a step ahead when getting to the actual data transfer. Instead of exchanging the data with the host managing (i.e. "owning") the storage device we would greatly benefit from attaching a disk or tape drive directly to a network, hence enabling it to deliver the data directly to the application requesting it (i.e. 3rd party transfer). Enabling standards and technologies, namely iSCSI, exist and products start to be widely deployed.

Combining the aforementioned arguments, CMS computing and many other data warehouse applications would greatly benefit from a scalable storage architecture that:

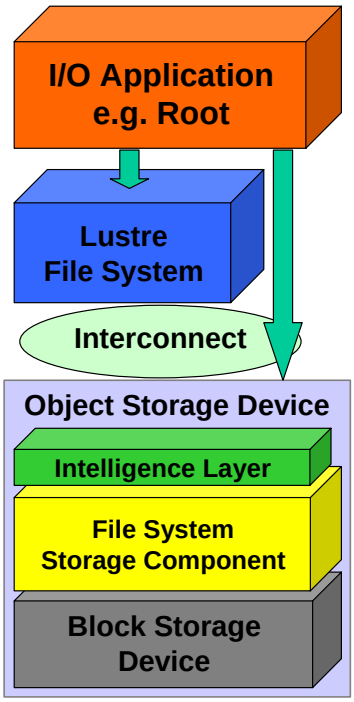
- Allows access granularity on the object level, hence:
  - Relieving any managing instance from inefficient micro-management
  - Avoids remapping of physics objects stored in files for further physics data analysis
- Allows 3rd party transfer in between applications and storage devices over a commodity network (e.g. GigE)
- Greatly simplifies the design and implementation of Hierarchical Storage Managers
- Has robust and usable secondary characteristics – for example good authentication and authorization, fault characteristics, commissioning characteristics.

As high performance, secure and affordable storage systems are vital to the success of the CMS program we recommend to spend the described effort, to be jointly carried out with prominent partners from the storage industry, to investigate the properties of OSD and associated technologies (e.g. Lustre file system).

## Proposed Project

The main investigation will focus on exploiting the capabilities of an OSD environment in order to meet the functional and performance requirements of the CMS application. Feasibility testing and evaluation will be done on this system in order to understand how such an OSD storage system could be scaled in size, performance, at appropriate cost over time to accommodate full-scale CMS experiments through the year 2010. Emphasis will be put on satisfying the research community needs and requirements for such a system.

The following diagram shows a ROOT I/O-based "application" that can access the Object Storage Device either directly or through the Lustre file system. The API from ROOT I/O to the Lustre file system is standard POSIX (open, close, read, write, fcntl). The API from ROOT I/O to the Object Storage Device is the OSD protocol.




The interconnection in this implementation will be Gigabit Ethernet although it is possible to use any interconnect such as Fibre Channel, Quadrics, InfiniBand, HyperTransport, Rapid I/O, …etc. This is useful when incorporating compute nodes with storage nodes in a bladed environment.

The "File System Storage Component" layer is internal to the Object Storage Device. This layer manages the objects on the actual physical block storage devices.

The "Intelligence" layer is included for completeness. Although not part of this project, this layer will make it possible to perform more complex functions such as "search and retrieve data based on a set of criteria".

The OSD protocol is still a very new protocol that is being worked out in the OSD Technical Working Group in SNIA. Because of this there are extremely few working examples of the Object-based Storage Device and an application that use such a device.

The xx-month goals of this project will be the following:

- Engage a few CMS researchers in developing an in-depth understanding of the data formats, data placement issues, and data access patterns that would benefit most from OSD
- Design and build a demonstration system at FNAL, using the Panasas Active Scale Storage Cluster technology, that will address issues mentioned above
- Compare and contrast the existing approaches to the data handling problem(s) to the OSD demonstration system
- Identify areas of improvement within the OSD demonstration system that would be appropriate for use in a full-scale production environment
- Investigate how this system will scale up for production use including a detailed cost, performance, and longevity analysis
- Investigate how this system would deploy and integrate into the existing FNAL infrastructure
- Compare the security implemented in the Luster OSD implementation with FNAL strong authentication requirements, and investigate other usability issues
- Generate a set of recommendations with respect to moving forward (or stopping) research and development of OSD-based systems for data handling

# Project Plan

This project plan consists primarily of the four basic phases defined in the Executive Summary. These are:

- Develop an understanding of the CMS storage and define data requirements for a demonstration system
- Demonstration System Design and Construction – Design and build a demonstration system based on the requirements
- Demonstration System Characterization – Run the demonstration system in a small-scale production mode, investigating its features in terms of performance and functionality
- Capabilities Comparison – Compare the strengths and weaknesses of current block-based and NAS-based storage systems to those of a mature OSD system.
- Improvement Analysis – Analyze constraints of current systems that OSD can potentially relax by using a prototype system to be constructed at FNAL for this purpose. Summarize potential improvements of mature OSD systems, but unavailable for study in the prototype.
- Production Scale Analysis – Write a report on what it would take to scale the demonstration system up to a full-scale production system.

Each of these phases is described in more detail below.

## *CMS Storage and Data Requirements*

Requirements gathering is necessary in order to define what needs to be demonstrated which will in turn drive how the demonstration system should be built.

### *Demonstration System Design and Construction*

This phase will synthesize the CMS storage and data requirements into a set of system requirements. The system will then be deployed and constructed over a period of about x months. During this time the actual demonstration needs to be defined.

### *Demonstration System Characterization*

Once the system is built we need to characterize the demonstration system in terms of its functional and performance capabilities and limitations.

### *Capabilities Comparison*

- Define the capabilities such a system should possess.

### *Improvement Analysis*

Analyze constraints of current systems that OSD can potentially relax, and would be studied in a prototype system.

Examples of this include

- Development of synthetic use cases, and plans for measurement and analysis of use of the system at different object grain sizes: for example:
  - physics object level (about 10**6 objects/gigabyte),
  - event level (about 10**3 objects/gigabyte)
  - aggregated level (about 1 object/gigabyte).

- Identification of useful fnctl() and non-posix I/O features that would support high bandwidth access to small objects that might be on disk-based OSD devices.

- Development of use cases identifying useful but missing functionality in OSD devices and file systems. Example use case: Fine grained, but high rate retrieval of data at the object level, based on bulk requests (i.e. a "better readv()")

### *System Demonstration*

The general requirements will also be used to develop a compelling demonstration that will lead to a far better understanding of the OSD-based solution to the FNAL data problem.

### *Production Scale Analysis*

Given the demonstration system and its functional, performance, and cost characteristics, determine how this solution will scale to meet full-scale production needs of FNAL as well as the other labs involved. It is also important to address specific issues related to the deployment and integration of OSD into the FNAL infrastructure.

## Milestones and Schedules

The following are estimates on how long it will take to address each of the tasks below. These times are stated in calendar weeks. Many of these tasks overlap one another and some can be done in parallel given appropriate staffing.

| What | Weeks | Deliverable | Due Date |
|---|---|---|---|
| Requirements gathering | 2 | Requirements document | |
| | | | |
| Problem Solution Space | 2 | Solution Space Document | |
| | | | |
| Demonstration System Specification | 2 | System bid specification | |
| | | | |
| Get/deploy System | 4 | N/A | |
| | | | |
| Construct System | 4 | Working system with 20 TB | |
| | | | |
| Demonstration Requirements | 2 | Demo Requirements document | |
| | | | |
| Demonstration implementation | 36 | Working demonstration system using OSD | |
| | | | |
| Capabilities Comparison | 6 | Capabilities Comparison doc | |
| | | | |
| Improvement Analysis | 6 | Improvement Analysis doc | |
| | | | |
| Scaling estimates | 6 | Scaling document | |
| | | | |
| Demonstration Analysis | 8 | Conclusions and recommendations document and presentation | |
| | | | |
| | | | |
| | | | |
| | | | |

Demonstration Requirements defines "'what" needs to be demonstrated.
Demonstration Implementation defines "how" the demonstration will be implemented and "how" the ideas will be demonstrated.

# Appendix A – Requirements

Requirements should be gathered from the:

- User community who will be accessing the system
- Administrators who manage the system and the data on the system
- FNAL Management who pay for this system and understand its part in the larger picture

The requirements gathered should include:

- System components – The type of computers, storage devices, networks, operating systems, file systems (local and shared/distributed), grid interfaces, scientific I/O packages, and other major software components whether they are commercially available, open source, or specific to FNAL.
- Performance metrics and expectations to be studied:
    - Bandwidth requirements from the storage to the servers, from the servers to the client computer systems, and to the user application
    - Latencies from the storage subsystem, through the networks, and to the user application
    - Transaction rates or throughput as viewed across the system rather than from a specific user's point of view

- Security – Requirements at the storage subsystem level, the network levels (SANs, LANs, and WANs). This includes authorization, authentication, encryption in flight as well as encryption at rest. Secure deletion/erase requirements should also be included even if there is no requirement.
- Capacities
    - Disk storage capacity of the initial test system (on the order of 10-20TB)
    - Capacity of any tertiary storage subsystems
    - Number of objects
    - Solid-state (RAM) on the storage systems including the Object-based Storage Device "engines" as well as caches on the disk arrays if these types of devices are employed

- Functionality – What are the primary functions that the storage system should be able to do including search and retrieve, backup, HSM, recovery, …etc. What other functions can the storage system do to offload processing from the compute farm?
- Reliability, Data Availability, and Serviceability Expectations – what is acceptable data loss? How is data availability measured – milliseconds, seconds, minutes, hours, days, weeks, fortnights? Who builds and maintains this system both at demonstration and production scales?
- Redundancy and Recovery
    - Sparing policies
    - Equipment replacement policies
    - RAID Levels - RAIDX N+1, what is X and what is N

- Scalability
    - Capacity
    - Bandwidth
    - Transaction rate
    - Connectivity
    - Geographic
- Cost – What is reasonable or practical
- Data Migration/Archiving capabilities
- Life expectancy
    - Systems
    - Data
    - Access to data

Phasing – time scales for Build-out, Production use, Phase-out